View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

Triune Continuum Paradigm and Problems of UML Semantics

Andrey Naumenko, Alain Wegmann
Laboratory of Systemic Modeling,
Swiss Federal Institute of Technology – Lausanne.
EPFL-IC-LAMS, CH-1015 Lausanne, Switzerland
andrey.naumenko@epfl.ch ; alain.wegmann@epfl.ch

Abstract

We present the results of our research that is positioned in the domain of system modeling. In particular, we present an object-oriented paradigm that provides a logically rigorous and complete theoretical base for various existing object-oriented frameworks. The strong points of the paradigm are presented by demonstrating how the paradigm can resolve a number of existing problems of the Unified Modeling Language (UML). The analysis of these problems and the proposed paradigm-based solutions represent an original research approach towards software systems modeling; the research approach is based on Russell's theory of types and on Tarski's declarative semantics theory. The paper advances the current state of research in software systems modeling frameworks in general and the state of UML research in particular.

1. Introduction

This paper targets two principal goals. The **first goal** is to introduce to readers the Triune Continuum Paradigm. This paradigm was originally defined in [7]; essentially it is a logically rigorous, internally consistent, complete and formally presented theoretical base for the conceptual organization of modern object-oriented frameworks that are used for system modeling in different contexts (e.g. in software development, in business modeling, enterprise architecture etc). This paradigm is an important contribution to the system modeling domain because currently none of the prevailing system modeling frameworks has a satisfactory formal theoretical foundation.

Considering the evolution of the software systems modeling languages, methodologies and tools through the last decade, among the numerous existing modeling techniques we may single out the Unified Modeling Language (UML). UML is a proposition of the Object Management Group (OMG) that emerged from the integration of different industrial practical experiences and became an influential phenomenon in the system modeling. As a matter of fact, due to the multiple efforts of different interested parties, UML has gained a relative domination over the other modeling techniques in the current industrial practices. This is why we decided to introduce the Triune Continuum Paradigm by positioning it in relation with UML, namely by noting and analyzing a number of UML problems and by explaining their concrete solutions, which are based on the proposed Triune Continuum Paradigm.

Thus the **second goal** of this paper is to advance the current state of the UML research, in particular of the research that concentrates on UML semantics.

The Triune Continuum Paradigm [7] is an original research finding that is based not only on solid logical foundations (such as Russell's theory of types [11] and Tarski's declarative semantics theory [12]) but also on the philosophical and natural science foundations (e.g. it features an extension of the traditional Minkowski's spatiotemporal reference frame [4] for the case of general system modeling). Thus with the aid of the paradigm we can present a non-traditional for software engineers view on the software systems modeling. We consider the originality of the paradigm-based view on software systems modeling to be a particularly important contribution of our research, because such an original view makes a difference in relation with the traditional views; it complements them, expanding the possibilities for the constructive evolution of software modeling languages, methodologies and tools.

In particular, as it will become clear from the analysis presented in this paper, our view on UML is complementary to the activities of OMG and different workgroups (such as U2 Partners, pUML, 3C UML) collaborating with OMG and submitting proposals for UML specifications (in particular for the specifications of UML 2.0 [1], [2], [13]). This is a real complementarity because UML interests us from a different perspective that is predefined by our goals which differ from the goals of

all known to us UML contributors. In particular, as we see it, all the aforementioned groups have one common characteristic: they are trying to contribute to the definition of the next version of the language. The results of our work are complementary because we do not intend to design a new version of UML, instead we are presenting a paradigm that can serve to different existing modeling frameworks (and, in particular, to UML) by providing its rigorous theoretical foundations to support the conceptual organizations of these frameworks.

Obviously, a paradigm (like Triune Continuum Paradigm) should not be a part of the specification of a modeling language; instead the language definitions can either conform to the paradigm or not conform to it. In the case when a language conforms to a paradigm, the paradigm would assure the theoretical soundness of the language. In the case when a language does not conform to any theoretical paradigm, there is no assurance for the language users; and any success or failure resulting from the language use is a matter of chance rather then a matter of founded reason.

Different languages may conform to the same paradigm. Hence in the particular case of Triune Continuum Paradigm, UML is just one of the modeling frameworks to which it can be related. So our work is absolutely independent from OMG or any other interested party. And our results, relating the paradigm with UML, are based on the independent analysis of the UML specifications.

To avoid the potential influences of "non-official" research results (the results of UML-related research that were not approved by OMG and those that are not a part of UML), in this paper we analyze the official UML specifications and nothing but the UML specifications that are interpreted as they are written in the latest official version of UML [9]. With our research results we do not intend to change UML; our intention is to attract the software engineering and business modeling communities' attention to some of the UML-related facts that otherwise pass unnoticed.

Our research approach is justified by the fact that in its current state UML does not conform to any theoretical paradigm. The explained complementarity of our work with respect to the dominating UML research threads makes our results particularly interesting for the attention of UML modeling community.

This paper is organized as following. Section 2 will present a summary of the Triune Continuum Paradigm including an overview of its advantages for system modeling and a presentation of its fundamental features. Section 3 will introduce three of the crucial problems of UML metamodel. Section 4 will analyze these problems and demonstrate their negative impact on the current UML terminology. Section 5 will present the three respective solutions, which are based on the Triune Continuum Paradigm, and explain the potential impact of these solutions on the future versions of UML terminology. Section 6 will present the conclusions which follow from our research experience with UML specifications, with their problems, and with the respective solutions that are based on the Triune Continuum Paradigm.

2. Summary of Triune Continuum Paradigm

This section will introduce the Triune Continuum Paradigm, in particular:

- the overview describing its most important functionalities that can be advantageous to the paradigm users;
- its main features, namely the most fundamental of the theoretical foundations which contribute to the internal organization of the paradigm;
- the context of its applications, namely the domain in which the paradigm is defined and can be used.

2.1. Overview of advantages

For the sake of presentation let us repeat here the first paragraph from the introduction. As it was said, the Triune Continuum Paradigm [7] is a logically rigorous, internally consistent, complete and formally presented theoretical base for the conceptual organization of modern object-oriented frameworks that are used for system modeling in different contexts (e.g. in software development, in business modeling, etc). We see this paradigm as an important contribution to the system modeling domain, because currently none of the prevailing system modeling frameworks has a satisfactory formal theoretical foundation.

The absence of theoretical foundation for modeling frameworks leads to the practical application experiences where modelers are constrained to be guided by chance and not by understanding of the fundamentals of systems analysis. And sometimes the chance fails them, which leads to the incorrect project specifications and in the end to the project failures. The Triune Continuum Paradigm fixes this problem of theoretical foundation.

The paradigm defines a rigorous and at the same time flexible metamodeling structure. This structure allows the definition of formal ontologies for various specific object-oriented frameworks, for example, as it is presented in [7], for UML or for RM-ODP [3] (RM-ODP stands for "Reference Model of Open Distributed Processing", which is an ISO/ITU standard for modeling of distributed systems). Thus different existing frameworks, like UML or RM-ODP, can benefit from the logical rigor, internal consistency, interpretation coherency, formal presentation and solid theoretical foundations of the defined paradigm.

Adoption of this paradigm allows the resolution of crucial problems existing in these different object-oriented frameworks.

Some of the existing system modeling frameworks (e.g. UML) appeared as an integration of the best modeling practices. The paradigm doesn't repudiate the practical experience that was gathered by these different frameworks, but fixes its inconsistencies and complements it supporting with logically rigorous theoretical foundations. Therefore the paradigm brings a significant constructive potential to the evolution of modern system modeling frameworks. This potential could be realized if people responsible for the design of modeling languages and tools would heed the proposed paradigm.

A concrete case of the paradigm application is formally presented and realized in a computer-interpretable form in [7] on the example of ontology describing the RM-ODP conceptual framework. This particular application of the paradigm realizes an important result that was officially targeted by the ISO/ITU standardization but was never achieved previously: a single consistent formalization of the RM-ODP standard conceptual framework. This formalization presents a concrete example of formal ontology for general system modeling.

2.2. Main features

In its essence the Triune Continuum Paradigm is an original research finding, which is constructed as a synthesis of the three theoretical foundations: of the Tarski's declarative semantics theory, of the Russell's theory of types and of an innovative solution defining a special observer-relational frame of reference based on the original notion of Triune Continuum.

Let us briefly present the roles that each of these three theoretical foundations plays in the internal organization of the paradigm.

2.2.1. Tarski's declarative semantics. The Tarski's declarative semantics theory [12] was proposed by Alfred Tarski in 1935. It is used in the paradigm to define formal relations between the subject that needs to be modeled and the possible models of this subject. The relation between the subject of modeling interest and its model is done by the modeler, who has the modeling interest with regard to the subject and who produces the model. The Tarski's theory suggests modelers to adopt an unambiguous way of the definition of this relation: an explicit one-to-one mapping between the modeler's perceived conceptualization of the subject of modeling and the representation of this conceptualization in the model. In this way the theory defines Tarski's declarative semantics.

The Tarski's theory shows that:

- If the declarative semantics <u>are adopted</u> then:
  - ⇒ if different modelers *agree* on the conceptualization of a subject of modeling then:
    - the modelers can formally compare their respective models representing this subject.
  - ⇒ if different modelers *do not agree* on the conceptualization of a subject of modeling then:
    - the modelers cannot compare their models in a logically rigorous way.
- If the declarative semantics <u>are not adopted</u> then:
  - ⇒ the modelers cannot compare their models in a logically rigorous way.

Thus, as assured by the Tarski's theory, modelers have a possibility to argue formally about their models in their community. And this possibility exists only in the case when the modelers both adopt the declarative semantics and agree on the conceptualization of subjects of their modeling interests.

Principles of Tarski's declarative semantics make an integral part of the Triune Continuum Paradigm, thus in a concrete application of the paradigm (e.g. a concrete modeling language) the paradigm assures the coherency in different interpretations of subjects of modeling interest.

2.2.2. Russell's theory of types. The Russell's theory of types [11] was defined by Bertrand Russell in 1908. This theory is used in the Triune Continuum Paradigm to ensure the internal consistency of the metamodeling structure proposed by the paradigm.

The metamodeling structure is one of the important features of the paradigm; its importance is explained by the fact that this structure should shape the metamodels of concrete system modeling frameworks that would adopt the Triune Continuum Paradigm for their concepts. Because of this importance the metamodeling structure needed a solid theoretical support. Thus the purpose was to determine an appropriate theory; and so, several structural constraints were defined from the outset of the paradigm definition. In particular:

- the metamodeling structure should have been *rigorous* – to thoroughly define the precise application contexts for the different concepts that could potentially make use of this structure;
- at the same time, the structure should have been *flexible* – to give a possibility for its adoption by the diverse range of already existing system modeling frameworks.

A solution for the theory that would obey these defined requirements was found in the Russell's theory of types.

The Russell's theory of types defines a structure of propositions that can be used in the logically rigorous constructions in a language to avoid the famous Russell's paradox [5]. The metamodeling structure of the Triune Continuum Paradigm was defined in [7] adhering to the structure of propositions introduced by Russell. In

particular for the construction of logically rigorous statements in a language Russell introduced (see [11]):

- individuals: "*We may define an individual as something destitute of complexity; it is then obviously not a proposition, since propositions are essentially complex.*"
- first-order propositions: "*Elementary propositions together with such as contain only individuals as apparent variables we will call first-order propositions.*"
- higher-order propositions (the second-order propositions, the third-order propositions, etc.): "*We can thus form new propositions in which first-order propositions occur as apparent variables. These we will call second-order propositions.*"

For the metamodeling organization of concepts used in a model, in correspondence with the Russell's theory, [7] defines:

- Model Elements (MEs) – direct analogs of the Russell's individuals. Model Element is the most general term referring to any element of the model. As well as the Russell's individuals, MEs are "*destitute of complexity*". This means that Model Elements, considered without the propositions associated to them, do not exhibit any particular information.
- Basic Modeling Concepts (BMCs) – direct analogs of the Russell's first-order propositions. In a model BMCs characterize Model Elements in the same way as the first-order propositions characterize the individuals in the Russell's theory of types.
- Specification Concepts (SCs) – direct analogs of the Russell's higher-order propositions. In a model SCs characterize BMCs in the same way as the higher-order propositions characterize the first-order propositions in the Russell's theory of types.

Thus the Russell's theory assures the necessary rigor of the paradigm's metamodeling structure. Also this structure is potentially flexible, because no particular constraint is given for the definitions of concrete BMCs and SCs.

2.2.3. Triune Continuum. To realize the potential flexibility of the defined metamodeling structure it was decided to define a minimal set of BMCs; the set which would be necessary and sufficient for a complete representation of the general system modeling scope on the most abstract level. Such a solution would allow different existing system modeling frameworks to place their modeling concepts as specializations of these BMCs (that is, as the concrete SCs) in the Triune Continuum Paradigm, and hence to adopt the paradigm with its solid theoretical foundations.

An original innovative solution was designed [7] to define and justify this minimal necessary and sufficient set of Basic Modeling Concepts. The solution provides to modelers a special observer-relational frame of reference. This frame of reference is one of the most important foundations of the paradigm; it is defined in [7] as a philosophically supported generalization of fundamental frameworks of natural science. In particular:

- in the classical (Newtonian) mechanics the observer-relational reference frames exhibit the relational nature in space, while time and material objects remain invariant for different observers;
- in the relativistic mechanics the observer-relational reference frames exhibit the relational nature in space and in time, while material objects remain invariant for different observers;
- in the Triune Continuum Paradigm the observer-relational reference frame exhibits the relational nature in space, in time, and in constitution of models that represents different subjects of modeling (including material objects) in the models. So, representations of material objects are observer-relational here.

The defined frame of reference is based on the original notion of Triune Continuum. This name owes to the triune essence of three continuums which, as shown in [7], are necessary and sufficient to represent the general system modeling scope. The three defined continuums are:

- the spatiotemporal continuum (introducing the space-time in the models);
- the non-spatiotemporal continuum (introducing the constitution of models);
- the information continuum (emerging as the information about the mutual relation of the space-time and the constitution of models).

The three continuums predetermined the necessary and sufficient set of Basic Modeling Concepts. The essential BMCs are: *space* and *time intervals* (belonging to the spatiotemporal continuum), *object* and *its environment* (belonging to the model constitution continuum), *state* and *action* (belonging to the information continuum and representing respectively static and dynamic information about objects and their environments in space-time).

2.3. Context of the paradigm application

The Triune Continuum Paradigm was defined as a General System Modeling theory[1]. This means that the paradigm is relevant and useful for modeling in all the different contexts of General System Modeling, and, in particular, in the context of software systems modeling. Hence different software systems modeling frameworks (and, in particular, UML) can benefit from the advantages of the paradigm that were presented in Section 2.1.

[1] The readers can see [7] for the formal definition of the General System Modeling domain.

3. Identification of problems in UML metamodel

When developing any modeling language, the language designer needs to define a scope of the language applications and then to define a set of modeling concepts that would be necessary to represent the defined scope. For the language to be useful in modelers' community practices, the modeling concepts need to have clear, logically structured and consistent semantics. In other words, the better structured the semantics are, and the less internal inconsistencies they have – the more useful the language for the modelers that are interested in representing the identified modeling scope.

Unified Modeling Language (UML) was designed by the Object Management Group (OMG) as "*a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems*" ([9] section 1.1). This identifies the scope of UML applications.

The experience of modeling practices in modern industries shows that UML is found useful by modelers. The amount of modeling projects that use UML, the amount of books written about UML and the number of software tools that support UML are large in relation with the analogous practical achievements of other modeling languages. This proves that UML in its current state is more practical then other modeling solutions, although it doesn't mean that there are no problems with the current state of UML.

Consistently with the scenario explained in the first paragraph of this section, the UML specification [9] introduces a set of modeling concepts to represent the identified modeling scope. Section 2 of the specification defines UML semantics for these concepts.

The first problem we can identify is that these UML semantics are considered to be complex and difficult to understand by many modelers. OMG itself in its article "*Introduction to OMG's Unified Modeling Language (UML™)*" [10] confirms this, saying that the UML specification [9] is "*highly technical, terse, and very difficult for beginners to understand*". This situation can be improved by analyzing the current state of UML semantics, understanding the reasons that cause its complexity and by proposing a better organization of semantics for modeling concepts. In particular, we will show that the Triune Continuum Paradigm, by introducing a logically precise and internally consistent semantics structure that is based on Russell's theory of types [11], makes a positive difference in relation with the absence of such a structure in the current UML semantics. The explicit presence of such a structure helps to understand how the modeling concepts should be used in practice, whereas its absence creates numerous possibilities for confusions in practical applications of modeling concepts.

While performing the analysis of the current UML semantics we can localize **the second problem**. Specifically that current UML semantics are very ambiguous in presenting relations between models constructed using the language on one side and the subject that is being modeled on the other side. This is an important problem, because even an internally consistent model will not have much of practical sense when its relations with the subject that it is supposed to represent are undefined. This situation with UML can be improved with the aid of the Triune Continuum Paradigm through the introduction of a coherent and unambiguous set of modeling concepts definitions expressing a kind of Tarski's declarative semantics [12] for the mentioned relations between the model and the subject of modeling.

The third problem of the UML semantics, which we will consider in this paper, is the absence of any justifications in the UML specification that would explain why the presented set of UML modeling concepts is necessary and sufficient to represent the UML modeling scope. Without these justifications, the UML theoretical value is significantly diminished, since in this situation the language cannot prove the reasonableness of its ambitions to represent its modeling scope. In the Triune Continuum Paradigm, the introduction of the set of modeling concepts is supported by solid philosophical and natural science foundations providing such kind of justifications.

4. Problems analysis based on the foundations of UML semantics

As we can see from Section 3, all three identified problems are related to the non-optimal semantics definition. Let us look at foundations of the UML semantics in order to localize the chapters in specifications from where the mentioned problems originate. The UML specification [9] in section 2.4 introduces the semantics Foundation package: "*The Foundation package is the language infrastructure that specifies the static structure of models. The Foundation package is decomposed into the following subpackages: Core, Extension Mechanisms, and Data Types.*" Analyzing the specification further we see for these three packages:

- Core: "*The Core package is the most fundamental of the subpackages that compose the UML Foundation package. It defines the basic abstract and concrete metamodel constructs needed for the development of object models.*" [9], section 2.5.1.
- Extension Mechanisms: "*The Extension Mechanisms package is the subpackage that specifies how specific UML model elements are customized and extended*

with new semantics by using stereotypes, constraints, tag definitions, and tagged values. A coherent set of such extensions, defined for specific purposes, constitutes a UML profile." [9], section 2.6.1.

- Data Types: "*The Data Types package is the subpackage that specifies the different data types that are used to define UML. This section has a simpler structure than the other packages, since it is assumed that the semantics of these basic concepts are well known.*" [9], section 2.7.1.

Thus we can conclude that the three identified problems originate from the Core package of UML. Consequently it is on this package that we will focus our further consideration.

4.1. Problem 1: Structural chaos of UML semantics

Let us now concentrate on the first of the identified problems: the absence of a consistent structural organization of UML metamodel that leads to practical difficulties in understanding semantics for particular modeling concepts, as well as to the difficulties in understanding semantically allowed application contexts for a particular modeling concept.

As it is presented in Figure 2-5 from the Core package specification [9], the most general concept in the UML metamodel is called "Element". It is defined ([9], section 2.5.2.16) as following: "*An element is an atomic constituent of a model. In the metamodel, an Element is the top metaclass in the metaclass hierarchy. It has two subclasses: ModelElement and PresentationElement. Element is an abstract metaclass.*" Thus any atomic constituent of a UML model can be called as UML element.

As it is presented in the diagrams 2-5,6,7,8,9 of the UML specifications [9], all the other modeling concepts are specializations of "Element". This defines a flat structure for the UML metamodel, where any of the concepts can be used as UML elements. And even if the elements obviously belong to different semantic categories (for example, "Operation" and "Class"), there is no explicit categorization defined to help a modeler to understand which concepts should be used in which context.

We may notice an introduction of "abstract" and "concrete" constructs categories in section 2.5.1 of the UML specification: "*Abstract constructs are not instantiable and are commonly used to reify key constructs, share structure, and organize the UML metamodel. Concrete metamodel constructs are instantiable and reflect the modeling constructs used by object modelers (cf. metamodelers). Abstract constructs defined in the Core include ModelElement, GeneralizableElement, and Classifier. Concrete constructs specified in the Core include Class, Attribute, Operation, and Association.*" However, this categorization becomes quite confusing if it is compared with the actual terms' definitions presented in the UML specifications. For example, "Association" is defined ([9], section 2.5.2.3) relative to "Classifier", which means that "Association" can be considered as both the abstract and the concrete construct. To summarize, the categorization of concepts into the abstract and the concrete constructs does not have a consistent implementation in the current UML specifications and cannot help modelers who would like to understand the possible application context for a particular modeling concept.

An approximate sketch of another possible categorization can be found in section 2.5.2 of UML specifications. The section introduces the figures 2-5,6,7,8,9 as following: "*Figure 2-5 on page 2-13 shows the model elements that form the structural backbone of the metamodel. Figure 2-6 on page 2-14 shows the model elements that define relationships. Figure 2-7 on page 2-15 shows the model elements that define dependencies. Figure 2-8 on page 2-16 shows the various kinds of classifiers. Figure 2-9 on page 2-17 shows auxiliary elements for template parameters, presentation elements, and comments.*"

So a reader could guess that "Backbone", "Relationships", "Dependencies", "Classifiers" and "Auxiliary Elements" are probably different categories of the modeling concepts. Unfortunately these pseudo-categories are neither defined in the relations between each other, nor in some other theoretical or practical application context. In addition, if we check the described figures, we see that the same modeling concepts (e.g. "Classifier" or "Relationship") are present at the same time in several of the diagrams. Thus a potential differentiation between the pseudo-categories is particularly difficult to understand.

We can conclude that the current UML specification of the Core fails to introduce a practically useful categorization of concepts that would define different application contexts for different conceptual categories. Unfortunately this problem cannot be solved by a simple adoption of some categorization for the currently existing UML concepts. This is due to the absence of any explicitly mentioned consistent strategy of concepts introduction by UML. In fact, judging from the specification, for us the strategy for the introduction of particular concepts remains obscure even on an implicit level. Surprisingly some concepts seem to appear without a significant justification whereas other conventional object-oriented terms are omitted.

For example, let us look at definitions of "ModelElement" and "PresentationElement", which are the two subclasses of UML element. We see that

"PresentationElement" is defined ([9], section 2.5.2.33) as "*a textual or graphical presentation of one or more model elements.*" Thus essentially a "PresentationElement" is a "ModelElement" presented in a textual or a graphical form. Here we may mention that, in general, a "ModelElement" from inside a model doesn't make sense to anybody or to anything if it is not presented in some form to somebody or to something who perceives the model in this form of presentation. Thus we may affirm that, in general, a "ModelElement", as soon as it is of interest to somebody or to something, is necessarily a "ModelElement" presented in some form. Thus, in fact, "PresentationElement" is a specialization of "ModelElement" where the forms of a possible presentation are known concretely (namely a textual and a graphical form). This specialization is the only value that is added to the semantics of "ModelElement" to obtain the semantics of "PresentationElement". Because of this minor significance of the added value, we may consider "PresentationElement" as not important enough to be a separate concept inside the UML metamodel. The elimination of "PresentationElement" if it is accompanied by the addition of the descriptions of possible ways of presentation inside the definition of "ModelElement", would simplify the metamodel without diminishing its value.

4.2. Problem 2: Absence of the declarative semantics in UML.

After having studied the complete UML metamodel, we can note that the UML specifications define explicitly only two concepts whose definitions are made by referring (relating) to the subject (system) that is being modeled. The first concept is "ModelElement", it is defined ([9], section 2.5.2.27) as "*an element that is an abstraction drawn from the system being modeled.*" The second of the two concepts is "Component", it is defined ([9], section 2.5.2.12) as following: "*A component represents a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces*". All the other concepts that constitute the metamodel are defined as parts of a UML model, **only** in the relations with each other and with the two mentioned concepts. That is, the definitions of all the UML metamodel concepts, with exception of the two mentioned, do not make reference to the subject that is being modeled. This semantics definition is not optimal.

Indeed as we said, only two concepts used in UML models are defined by a reference to the subject being modeled. More than that, the UML metamodel doesn't define why these two concepts and why only these two (and not some other) were designated for this purpose. This means:

a. that this choice of these two concepts does not have a tenable reason defined in the UML specification;
b. that UML specification does not define a tenable relation between a subject that needs to be modeled and its model.

The conclusion 'b.' is particularly important, because it means that for the UML concepts the specification does not define any kind of formal declarative semantics that were introduced by Alfred Tarski [12]. Indeed, Tarski's declarative semantics for concepts used inside models are supposed to introduce mappings between the agreed conceptualizations of a subject that is being modeled and the concepts inside its model. The UML metamodel never presents an agreed conceptualization of the subject of modeling. Thus the specification has no choice but to define modeling concepts exclusively in their interrelations inside the model. In the general case, this approach is not an optimal one for the following two main reasons:

1. The overall complexity of the relations between concepts in the UML metamodel is greater than it would have been if part of the concepts were defined in the relations with the subject of modeling. Indeed, the quantity of concepts is the same in both cases, but in the latter case some concepts would be defined in a self-sufficient way, whereas in the former, the corresponding concepts for their definitions will need relations to other concepts from the metamodel.
2. Since there is no tenable relation defined between a subject that needs to be modeled and its model and since there is not any agreed conceptualization of the subject, there cannot be a formal proof (such as in the case of Tarski's declarative semantics) that a given modeler's interpretation (that is a model) represents the subject of modeling in a logically consistent way[2]. In other words, in this case several mutually contradicting models can represent the same subject of modeling and all of them may be confirmed as adequate; or, from the other side, one single model may be related with the same degree of adequateness to different mutually incompatible subjects of modeling.

To illustrate the second point let us take Tarski's original example [12] for declarative semantics definition:

[2] Here we mean the logical consistency in an interpretation of subject of modeling. The internal consistency of a model (the model being a result of the interpretation) is a different subject. The internal consistency of a model can be ensured by the consistency of the UML metamodel, and to ensure the logical consistency of the interpretation, a kind of consistent Tarski's declarative semantics is necessary.

'It snows' is true (in the model) if and only if it snows (in the subject of modeling). So if we decide to take the subject of modeling where it snows, without the declarative semantics, then both 'It snows' and 'It does not snow' can be considered true in the model if it snows in the subject of modeling. From the other side, without the declarative semantics the model where "'It snows' is true" may represent equally well both the subject of modeling where it snows and the subject of modeling where it does not snow.

In the case of UML specification, as we said, there are only two concepts that make the direct relation to a subject of modeling: "ModelElement" and "Component". Parts of their definitions can be considered as introducing the declarative semantics. For example, according with [9], sections 2.5.2.27 and 2.5.2.1 we can write for "ModelElement": 'A ModelElement exists' is true in the model if and only if a subject of modeling ("system being modeled") is. And for "Component" according with [9], section 2.5.2.12 we can write: 'A Component exists' is true in the model if and only if there is a modular, deployable, and replaceable part of the subject of modeling ("of system"). But as we see, these definitions are too abstract: they do not give a possibility for a differentiation of modeling concepts, thus the choice of only these two concepts to be defined using the declarative semantics is not practical.

4.3. Problem 3: Absence of theoretical justifications for the UML metamodel to represent the targeted modeling scope

As we said, the third problem of UML semantics is the absence of any justifications in the UML specification that would explain why the presented set of UML modeling concepts is necessary and sufficient to represent the UML modeling scope. This problem can be considered as natural for the current state of UML because, from its outset, the language was constructed by OMG as a result of the integration of the best existing industrial modeling practices, but these practices were never really linked with the existing scientific theories. Although the "best practices" strategy can be considered as an attempt at practical justification of UML, the theoretical justification was never defined in the language specifications and still needs to be provided.

Thus as we can see, this third problem of the UML metamodel is a theoretical problem, compared to the first two identified problems that are practical. So the third problem is important as soon as UML would pretend to be a well founded modeling technique. This can increase the social impact of UML and can be useful, for example, for the language standardization by some international standardization committee, which would normally assume solid scientific foundations rather then just results of practical experience to support the language.

4.4. Impact of the reviewed problems on the UML terminology

To emphasize the importance of the three reviewed metamodeling problems, in this subsection we will demonstrate their consequences on concrete examples. When talking about consequences of the metamodeling problems, we differentiate two levels of argumentation: metamodel and its realization. Indeed, for a generic metamodeling problem its produced effects become visible in a concrete realization of the metamodel. That is, in our case, to see the consequences of the three reviewed metamodeling problems we have to check the concrete definitions of concepts within the UML modeling framework.

A single metamodeling constraint can be relevant for a big number of modeling concepts, thus, naturally, a single metamodeling problem can cause multiple cases of problems with definitions of concrete concepts. The primary interest of this paper is UML metamodel and not a concrete realization of the UML terminology. So for our interests here we will just show a couple of examples that will demonstrate typical flaws of the UML terminology, the flaws that are effects of the three reviewed metamodeling problems. And the primary goal of this presentation is not to argue the concrete concepts of UML, but to show that the consequences of the reviewed problems are important and thus the problems should not be neglected by the UML modelers.

After having explained our inducements let us proceed with the presentation. Through the examples in this subsection we will see the following two problems of the UML terminology:

- Terminology Problem 1: Some of the fundamental UML terms are questionable, the reasons for their introduction are unclear (this problem is a direct consequence of the absence of theoretical justifications for the UML metamodel reviewed in Section 4.3 of this paper);
- Terminology Problem 2: The semantics of the UML terminology are undefined (this is the consequence of the two general problems of the UML metamodel reviewed in Sections 4.1 and 4.2, namely, of the structural chaos of UML semantics and of the absence of declarative semantics in UML).

4.4.1. Example: "Association" and "Classifier". This example will illustrate the first problem of the UML terminology, which is the questionable nature of some of the fundamental UML concepts. As we already mentioned this problem owes to the absence of theoretical

justifications for the UML metamodel reviewed in Section 4.3 of this paper. Because of this absence the reasons for introduction of the UML terminology are unclear and in some cases the currently existing terminology [9] can be easily challenged.

In the Core package of UML Semantics (see section 2.5 of [9]) UML specifications define, among others, two fundamental UML terms: "Association" and "Classifier". These two terms are relatively significant in the UML conceptual framework; this is why we decided to choose them for the presentation of our example.

In the UML metamodel, "*an Association is a declaration of a semantic relationship between Classifiers, such as Classes*" (see section 2.5.2.3 of [9]). Analyzing this definition as a general case definition of "Association", we can differentiate the following two cases as possible specializations of the general case:

- Case 1: Association is a declaration of a semantic relationship between classifiers done (seen) from the point of view of a classifier that participates in this relationship;
- Case 2: Association is a declaration of a semantic relationship between classifiers done (seen) from the point of view of a classifier that does not participate in this relationship.

Obviously, these two cases cover completely the possible classifier-related contexts in which the "Association" can be considered. Indeed, as "Association" is defined as "*a declaration of a semantic relationship between Classifiers*", a "Classifier" may either participate in the relationship or not participate in the relationship. The case where "Classifier" is irrelevant with regard to the relationship is excluded by the quoted definition. From the other side, under this definition a semantic relationship is always associated with one or more classifiers. The case when the relationship is associated with several classifiers can be seen as a specialization of Case 1 (we will present this specialization further as Case 1.1). Thus Case 1 and Case 2 are not only two possible specializations of the general case, but also one of the two cases is always necessary realized.

Of course, the two cases may have further specializations; for example, the following case:

- Case 1.1: Association is a declaration of a semantic relationship between classifiers done (seen) from the classifiers' supra-system point of view, when all the classifiers participating in the relationship belong to the supra-system.

Case 1.1 is a specialization of Case 1, where the participating classifier from Case 1 participates in the relationship by means of its internal sub-classifiers (and thus the participating classifier is called "supra-system" in Case 1.1). Thus Case 1.1 presents a supra-system viewpoint and also presents the situation where different sub-classifiers share the same declaration of the semantic relationship.

In fact, the different possible specializations of Case 1 and Case 2 are not important for our discussion. The important thing is that both in Case 1 and in Case 2 "Association" is a feature (property) that belongs to some concrete single classifier: in the first case – to the classifier that participates in the relationship; in the second case – to the classifier that does not participate in the relationship. Because Case 1 and Case 2 cover completely all the possible contexts in which the "Association" can be considered, we can affirm that in the general case "Association" is a feature of some concrete single classifier.

Let us turn our attention to the definition of "Classifier": "<…> *In the metamodel, a Classifier declares a collection of Features, such as Attributes, Methods, and Operations*" (see section 2.5.2.10 of [9]).

Looking at the definition of "Feature" we see (section 2.5.2.21 of [9]): "*A feature is a property, like operation or attribute, which is encapsulated within a Classifier. In the metamodel, a Feature declares a behavioral or structural characteristic of an Instance of a Classifier or of the Classifier itself. Feature is an abstract metaclass.*"

Other definitions relevant in the context of Classifier's definition include "Attribute", "Method" and "Operation":

- For "Attribute" in section 2.5.2.6 of [9]: "<…> *In the metamodel, an Attribute is a named piece of the declared state of a Classifier, particularly the range of values that Instances of the Classifier may hold.*"
- For "Method" in section 2.5.2.26 of [9]: "<…> *In the metamodel, a Method is a declaration of a named piece of behavior in a Classifier and realizes one (directly) or a set (indirectly) of Operations of the Classifier.*"
- For "Operation" in section 2.5.2.6 of [9]: "<…> *In the metamodel, an Operation is a BehavioralFeature that can be applied to the Instances of the Classifier that contains the Operation.*"

From these definitions we see that "Feature" in general, and "Attribute", "Method" and "Operation" in particular, are the concepts whose concrete definitions are essentially a declaration of semantic relationship between some classifiers. Indeed:

- For "Attribute" to have "*the declared state of a Classifier, particularly the range of values that Instances of the Classifier may hold*" we need to specify a relationship to those mentioned values; and, according to the "Classifier" definition, values should be considered as classifiers because they also "*declare a collection of Features*".
- For "Method" to have "*a named piece of behavior*" we need to define a relationship to the specification of constraints (e.g. spatiotemporal, structural, etc) that define this behavior; and, according to the

"Classifier" definition, these constraints should also be considered as classifiers because they also "*declare a collection of Features*".

So, essentially "Feature" is a declaration of semantic relationship between some classifiers, and thus "Classifier" is essentially a collection of "Associations".

We showed that "Association" is always a feature of some concrete "Classifier", and that "Classifier" is essentially a collection of "Associations". Thus we can challenge the existence of any reason to differentiate "Classifier" and "Association" as two fundamentally different concepts.

This result of our analysis of UML specifications is not surprising, because in conceptual modeling[3], in general, all concepts matter in relations. It is impossible to define any concept without defining a relationship. To define some concept one needs to define its relations to something else, to something that is not this concept. The set of relations defining a concept automatically defines the context in which it is relevant to consider the concept. Any concept makes sense in a context, without a context it just cannot be defined: cannot be differentiated because there is nothing to differentiate it from. So from the general point of view every concept is a relation.

Thus in conceptual modeling the term of relationship is too general to represent a meaningful modeling concept. So, from the conceptual modeling point of view it is not a good choice for UML to take "Association" as one of its basic modeling concepts. As we already mentioned, unfortunately, UML specifications do not provide the reasons for introduction of the defined UML terminology. Thus it is not clear whether these reasons exist or not at all; and therefore it is not possible to analyze such hypothetic reasons for their theoretical soundness, internal consistency and logical rigor. So, in its present state [9] UML terminology can be considered as baseless.

Obviously, this terminological problem would not exist if the necessity of the UML terms and their sufficiency for the UML scope representation were justified on the metamodeling level. Thus the problem of absence of theoretical justifications for the UML metamodel (that we analyzed in Section 4.3) has concrete negative impact on the current state of UML terminology and should not be neglected.

As it will be explained in Section 5 of this paper, the Triune Continuum Paradigm resolves the problem of theoretical justifications, and thus, it allows to define terminologies that are destitute of the presented problem of the UML terminology. For example, RM-ODP terminology [3], which obeys terminological requirements of the Triune Continuum Paradigm [7], doesn't have a modeling term that would be analogous to the UML "Association". While the RM-ODP "Object" can be considered analogous to the UML "Classifier".

[3] For the formal definition of the conceptual modeling domain see for example [7].

4.4.2. Example: semantics of "Object". In this subsection we will present an example that will illustrate the second fundamental problem of the UML terminology, which is the fact that for the vast majority of UML terms the semantics remain undefined by UML specifications [9]. In many cases the existing in [9] terms definitions hinder a meaningful consistent logical interpretation of the terms semantics. As we already mentioned in the beginning of Section 4.4, this is the consequence of the two general problems of UML metamodel reviewed in Sections 4.1 and 4.2, namely, of the structural chaos of UML semantics and of the absence of declarative semantics in UML.

Here we represent the most sorrowful of our research experiences with UML: after having done a detailed study of UML specifications [9] we have to affirm that with the UML terminology in its current state of definitions it is impossible to construct a reasonable modeling framework. Unfortunately, the definitions for terminology from the Core package of UML semantics as they are currently presented contain multiple logical contradictions that can only be resolved either with the complete absence of the terms interpretation or with a free meaningless interpretation for the terms.

As the concrete example we will present one of the key confusions in the UML terminology definitions from the Core package of UML specifications; this confusion is related with the word "object". Definitions of terms like "Class" ([9], section 2.5.2.9), "Flow" ([9], section 2.5.2.22), "Node" ([9], section 2.5.2.29), "Operation" ([9], section 2.5.2.30) are referring to "object", while "object" itself is not defined in the Core package, and what exactly was meant by "object" in these definitions remains impossible to deduce.

Let's look for instance on the definition of "Class": "*A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics. <…>*". From this phrase we can understand that "object" for the UML specifications is something that is supposed to have some semantics. Omitting for the moment the question about these concrete semantics definition, from the fact that some semantics are defined we may conclude that "object" is a modeling construct. That is, "object" is something which exists in the model and controlled by a modeler with the aim to represent in accordance with the defined semantics something from a universe of discourse, namely from the universe of discourse that is a subject (system) being modeled.

Further in the definition of "Class": "*In the metamodel, a Class describes a set of Objects sharing a collection of Features, including Operations, Attributes and Methods, that are common to the set of Objects. <…> A Class*

defines the data structure of Objects, although some Classes may be abstract; that is, no Objects can be created directly from them. Each Object instantiated from a Class contains its own set of values corresponding to the StructuralFeatures declared in the full descriptor. <…>". The first phrase from the last quote still supports the vision that object is a modeling construct, that is a part of model. However, the last part of the quote assumes that "Object" can be "*created*" or "*instantiated from a Class*". Referring to the definition of semantics for "Instantiation" ([9], section 2.5.4.5), we see: "*The purpose of a model is to describe the possible states of a system and their behavior. The state of a system comprises objects, values, and links. Each object is described by a full class descriptor. The class corresponding to this descriptor is the direct class of the object.* <…>". Here in opposition to the previous conclusions we may clearly see that "object" is a part of the system that is represented by a model (as well as "value" and "link", which would have analogous interpretation problems by the way).

So, in which domain "object" is?! The first half of the "Class" definition suggests that "object" is in the model, while the second half of the "Class" definition as well as the "Instantiation" definition suggests that "object" is in the system which is modeled. These are clearly two different domains, and they cannot contain the same constructs: the model is under the complete modeler's responsibility and control (which allows for the modeler's definitions of concrete formal semantics for the modeling constructs), while the system that is modeled is not under the modeler's control (which only allows for an experiential conceptualization of the system).

Thus the references to "object" in the analyzed definitions introduce contradiction. This concrete contradiction makes impossible a logical interpretation of the concerned definitions. Unfortunately this is just one of many negative examples that can be found in the UML specifications.

Let us return to the concrete definition of semantics for "object". Such definition cannot be found in the chapter 2 of UML specifications that is called "UML Semantics" and that as defined ([9], section 2.1.1) "*provides complete semantics for all modeling notations described in UML Notation Guide (Chapter 3)*". However, surprisingly, a semantic definition for "Object" can be found in UML Notation Guide ([9], section 3.39.1). It defines: "*An object represents a particular instance of a class. It has identity and attribute values.* <…>" Which when put in the same context with the previously quoted ([9], section 2.5.2.9) "*A class is a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.* <…>" makes the following construction: "An object represents a particular instance of a description of a set of objects that share the same attributes, operations, methods, relationships, and semantics." Does this pretend to be a tautology? We cannot know the answer since semantics of "instance" that is heavily used as a modeling concept in UML were never defined in UML specifications.

We may also note the question on whether "object" and "Object" from ([9], section 2.5.2.9) and "run-time physical object" from ([9], section 2.5.2.29) are three identical things or not; and if not, what the differences between them are…

These are just few of many analogous problematic cases. Other examples, such as the analogous analysis that would try to examine what UML specifications see by "instance" would show even more of despair. All these things make a very unpleasant research experience that practically shows that UML specifications in their current state present terms definitions which are self-contradictory and contain many baseless relations.

This terminological problem would not exist if UML metamodel had a well organized structure and, in particular, if the terms' declarative semantics were defined coherently, while preserving the logical differentiation between the subjects of modeling interest and the models of these subjects. Thus the two problems of UML metamodel that we analyzed in Sections 4.1 and 4.2 have concrete negative impact on the current state of UML terminology and they should not be neglected.

Here let us summarize Section 4 of the paper. The examples and their analysis that we presented in Section 4.4 demonstrate a regrettable situation with the UML terminology. As we explained, this regrettable situation owes to the three problems of UML metamodel that were analyzed in sections 4.1-4.3. Thus, by solving the three metamodeling problems we eliminate the source of the terminological flaws. The next section will present concrete solutions for the three metamodeling problems that we analyzed in sections 4.1-4.3.

5. Solutions for the identified problems of UML metamodel

In Section 4 we analyzed three crucial problems of UML metamodel. Here we present the three respective solutions that are provided by the Triune Continuum Paradigm. These solutions shape an alternative metamodel that being destitute of the analyzed problems allows to define a successful terminology and thus demonstrates the potential for improvement for the current state of UML.

5.1. Solution to problem 1: Categorization of concepts based on the Triune Continuum Paradigm

As we explained in Section 4.1, UML doesn't define any explicit logically consistent strategy for the

introduction of modeling concepts. In order to solve the problem of the structural chaos of UML semantics we can use a theoretical framework that defines such kind of strategy, for example the Triune Continuum Paradigm.

To define the metamodeling structure the Triune Continuum Paradigm takes the basic conceptual structure of the RM-ODP [3] standard (part 2: Foundations) and reinforces it by means of the strong theoretical foundations of Russell's theory of types [11], as well as by means of the structural principles of Tarski's declarative semantics [12].

As it was proposed in RM-ODP part 2 clause 6 that defines "Basic Interpretation Concepts" conceptual category, in the paradigm we call the subject of modeling (which is the subject that has some modeling interest to a modeler) as "Universe of Discourse". In RM-ODP, "Universe of Discourse" was constituted by entities (defined in [3] 2-6.1 as "*any concrete or abstract thing of interest*") and propositions that can be asserted or denied as being held for entities (defined [3] 2-6.2).

This notion of the "Universe of Discourse" organization is compatible with Russell's theory of types [11] defined by Bertrand Russell in 1908, that introduces individuals and propositions over individuals. In particular, [11] explains:

"*We may define an individual as something destitute of complexity; it is then obviously not a proposition, since propositions are essentially complex. Hence in applying the process of generalization to individuals we run no risk of incurring reflexive fallacies.*

Elementary propositions together with such as contain only individuals as apparent variables we will call first-order propositions. We can thus form new propositions in which first-order propositions occur as apparent variables. These we will call second-order propositions; these form the third logical type. [while individuals form the 1st logical type and the first-order propositions form the 2nd logical type] *Thus, for example, if Epimenides asserts "all first-order propositions affirmed by me are false," he asserts a second-order proposition; he may assert this truly, without asserting truly any first-order proposition, and thus no contradiction arises.*

The above process can be continued indefinitely. The (n + 1)th logical type will consist of propositions of order n, which will be such as contain propositions of order n - 1, but of no higher order, as apparent variables. "

Analogously, in the Triune Continuum Paradigm we have "entity" corresponding to Russell's "*something destitute of complexity*", because the only intrinsic meaning of an entity in the RM-ODP definition is to be "something" that can be qualified by means of propositions. An entity has no other meaning without the propositions associated with it. Thus, by mapping Russell's "individual" and "proposition" to the RM-ODP "entity" and RM-ODP "proposition", respectively, the Triune Continuum Paradigm uses Russell's suggestion in the context of the universe of discourse. This allows us to differentiate the propositions with regard to their subject of application:

– if a proposition is applied to an entity it is considered as the first-order proposition;
– if a proposition is applied to a proposition it is considered as the higher-order proposition.

Of course, in an application of these propositions there may be a situation when a higher-order proposition is applied on another higher-order proposition, which in its turn is applied on yet another higher-order proposition and so on, until the overall structure of the higher-order propositions is finally applied on the first-order proposition. Hence for simplification, we will refer to the combination of several higher-order propositions, which is applied on a first-order proposition, as a single higher-order proposition.

So the paradigm orders the entities and propositions that constitute a universe of discourse in agreement with the Russell's theory of types. Now we can look at models that should represent an arbitrary universe of discourse. A model is the place where modeling language constructs should be applied. Thus it is for the model part of the metamodel that the paradigm provides a useful structure of the categorization of concepts, which would explain the different contexts of practical applications for the concepts from different categories.

The paradigm suggests organizing the modeling concepts structure in such a way that there would be a straightforward correspondence between the model and the corresponding represented universe of discourse. That is, it suggests constructing a structure of concepts in the model in agreement with Russell's theory of types, which would correspond directly to the universe of discourse organization we presented earlier.

According to this suggestion, within the model we will be able to identify "Model Elements" that will be analogous to the Russell's "*individuals*" defined "*as something destitute of complexity*". Also, under this assumption, in the model we will have some concepts that are analogous to the Russell's "*first-order propositions*" (the paradigm calls them "Basic Modeling Concepts"), and some concepts – analogs of the "*higher-order propositions*" (called "Specification Concepts"). With this approach to the construction of a model it would be necessary to qualify "Model Elements" with the aid of "Basic Modeling Concepts", which in their turn could be qualified by means of "Specification Concepts".

Thus it is possible to define the correspondence between the conceptual categories from within the model and the entities and propositions that form the universe of discourse that should be modeled. The correspondence is defined as follows:

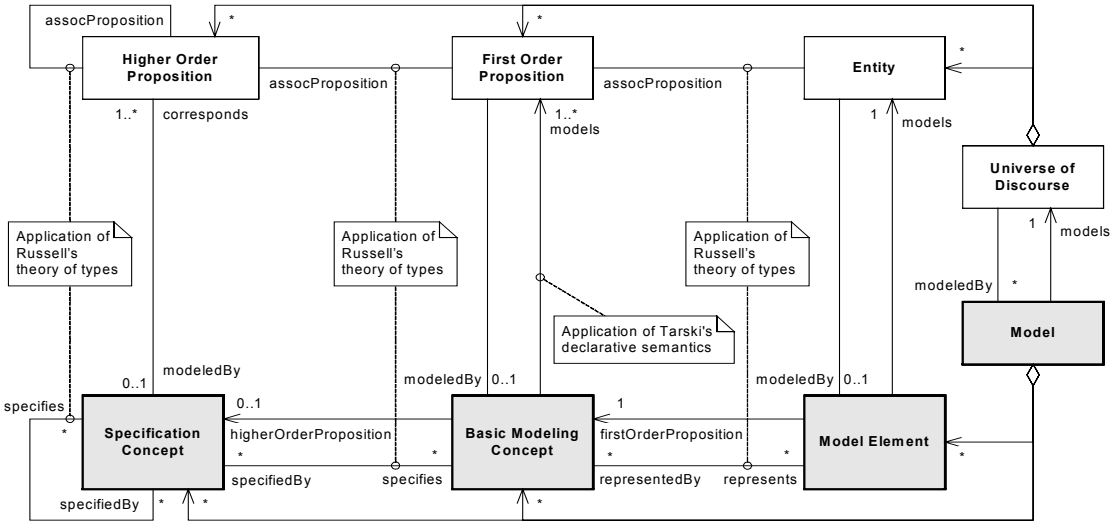

Fig. 1. Categorization of concepts in the metamodel proposed by the Triune Continuum Paradigm (UML diagram)

- *Entities* from the Universe of Discourse are modeled by *Model Elements* in the Model.
- *First-order Propositions* from the Universe of Discourse are modeled by *Basic Modeling Concepts* in the Model.
- *Higher-order Propositions* from the Universe of Discourse are modeled by *Specification Concepts* in the Model.

Hence, model elements are defined in the model as one to one counterparts to entities from the universe of discourse. Let us consider more closely the two other conceptual categories from within the model. As we explained, in correspondence with Russell's definitions, basic modeling concepts (essentially the first-order propositions) contain model elements as "*apparent variables*"; and specification concepts (the higher-order propositions) contain the basic modeling concepts as "*apparent variables*".

In fact, these two conceptual categories were introduced by RM-ODP specifications ([3] part 2, clauses 8 and 9); up to this point in our presentation we only saw that the Triune Continuum Paradigm reinforced logical justifications for this categorization with the support of Russell's theory of types and with explicit definitions of the application contexts for concepts from the two categories. For further explanation of the difference between concepts from the two conceptual categories in the paradigm we will use the principal structure of relations between a universe of discourse from one side and its model from the other side; this structure was defined by Alfred Tarski in 1935 for the introduction of his formal declarative semantics [12].

The basic modeling concepts set, as it aims to model the first-order propositions from the universe of discourse, should contain the concepts expressing the qualities that are considered as primary and intrinsic for the universe of discourse entities. This fundamental nature of the primary qualities belonging to the universe of discourse doesn't allow their modeling representations to be defined exclusively within the model. Hence the only possibility for a definition of the basic modeling concepts is to define them using Tarski's declarative semantics [12]: the semantics that define equivalence of an agreed conceptualization of the universe of discourse to a concrete concept in the model. The set of basic modeling concepts constructed in this way is the necessary, sufficient and limited set representing a limited amount of intrinsic qualities from the universe of discourse.

The set of specification concepts contains all the other concepts that can be found in models. These concepts aim to model the higher-order propositions from the universe of discourse; thus they do not represent the primary qualities of the universe of discourse entities and hence they do not need to have Tarski's declarative semantics for their definitions. So these concepts will be defined only in the relations between themselves and in the relations with the basic modeling concepts, but not in the relations with the universe of discourse. In a general case, the set of specification concepts is not limited because of the same quality of the higher-order propositions set. As new higher-order propositions can be constructed by applying one higher-order proposition on another, new specification concepts can similarly be constructed by applying one specification concept on another.

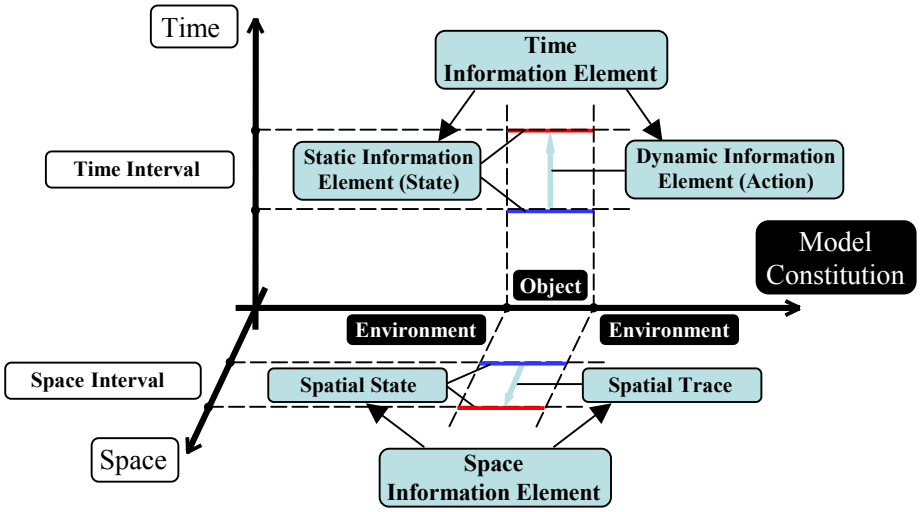

Fig. 2. Three-dimensional framework with the dimensions of "Space Continuum", "Time Continuum" and "Model Constitution Continuum", which allows for the emergent "Information" continuum.

So, it becomes clear that there is a significant semantic difference between the two conceptual categories. Basic modeling concepts are defined using Tarski's declarative semantics, but specification concepts are not. This is the consequence of the differences in their design purposes, which explains the clear difference in their corresponding applications within a model.

Additional details on this categorization can be found in [7]. Here let us present a UML diagram explaining the structure of the introduced categorization (see Figure 1)[4].

5.2. Solution to problem 2: Tarski's declarative semantics definitions for basic modeling concepts

The complete analysis of definitions for concepts from the basic modeling concepts category that was introduced in the previous subsection can be found in [7]. Here we will just briefly explain the overall structure of basic modeling concepts, and present this structure in the form of UML diagram.

Figure 2 presents the idea of general organization for the basic modeling concepts category. Essentially the set of concepts is determined by the consideration of the spatiotemporal conceptual continuum and the non-spatiotemporal conceptual continuum. The former represents in the model a space-time from the universe of discourse, and the latter represents in the model the non-spatiotemporal conceptual entities that constitute the universe of discourse. In correspondence with their ancestors from the universe of discourse, the former is presented by the Space and the Time dimensions on Figure 2, while the latter by the Model Constitution dimension. Being considered in the same context of a model, the two introduced conceptual continuums necessarily give birth to the third one that is essentially the Information about the Model Constitution within Space-Time. Detailed analysis of this approach can be found in [7].

By defining limiting points within Space-Time and Model Constitution dimensions we obtain concepts of Space Interval, Time Interval for the Space and the Time and concept of Object with the concept of its Environment for the Model Constitution. Also, with the definition of these limiting points we are able to consider Object and its Environment:

- at a single moment in Time, and thus to define the concept of Static Information Element (State) within the Information continuum;
- at an interval between two moments in Time, and thus to define the concept of Dynamic Information Element (Action) within the Information continuum.

Thus we obtain the structure of basic modeling concepts presented in the UML diagram from Figure 3.

In correspondence with the explanations from the previous section all the basic modeling concepts in the Triune Continuum Paradigm have formal definitions in the form of Tarski's declarative semantics. We recommend to check [7] for all these concrete definitions.

[4] In the diagram in Figure 1, in addition to all the explained particularities of the categorization structure, we also showed that a specification concept can specify any of the basic modeling concepts, and a basic modeling concept can be specified by any of the specification concepts. In fact this is true only for the generic specification concepts – the subcategory of specification concepts whose definition is beyond the scope of this paper and can be found in [7].

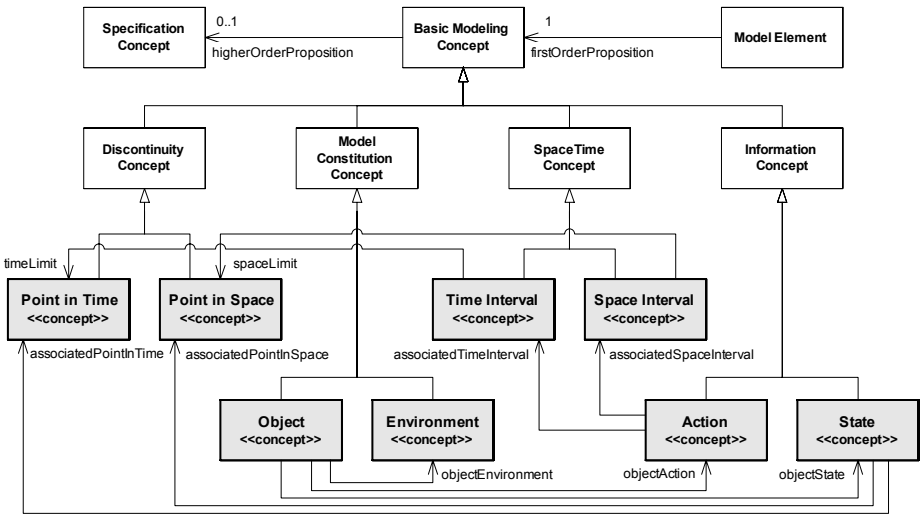

Fig. 3. Basic Modeling Concepts: Conceptual Specialization (UML diagram).

The definitions of basic modeling concepts, as well as the definitions for all the other concepts proposed in the paradigm, have much in common (and are even identical in many cases) with the definitions of corresponding concepts given by RM-ODP. With the aid of the paradigm we formalized the overall RM-ODP foundations framework ([7], [8]), including the basic modeling concepts part, using Alloy [6] formal description technique. Thus the basic modeling concepts semantics introducing a coherent set of Tarski's declarative semantics for relations between the concepts and the subject that is being modeled (universe of discourse) present a formally justified logical structure.

5.3. Solution to problem 3: Philosophical and natural science foundations of the Triune Continuum Paradigm

As we explained in the analysis of Problem 2 (Section 4.2), even for the choice of two modeling concepts that were linked in their definitions with the subject of modeling, UML specification does not define any tenable reason. And the set of modeling concepts that are defined using declarative semantics could be the very source of justifications for the ambitions to represent a given modeling scope with the modeling concepts of the language. Indeed, if the declarative semantics concepts cover all the possibilities of the agreed conceptualizations of the modeling scope then, the set of concepts can be considered as sufficient for the modeling purposes. And from the other side, the very set of declarative semantics concepts that would cover all the agreed conceptualizations of the modeling scope can be considered as necessary due to the necessity of the scope representation.

As the previous paragraph shows, the approach to the solution of the third indicated problem of UML metamodel is in the scientific justification of an agreed conceptualization of the modeling scope and in a formally defined unambiguous and logically consistent correspondence of the conceptualization to the modeling concepts that are designated to represent the conceptualization in the model.

The complete theoretical justifications of the universe of discourse conceptualization (that was introduced in the previous section to support the introduction of basic modeling concepts) can be found in [7]. Here we will just mention that:

- the possibility to define limiting points and thus discrete concepts within a conceptual continuum is justified by mereology that is a branch of philosophy studying whole-part relationships;
- the possibility to consider the constitution of models as a conceptual continuum that is independent with regard to the spatiotemporal continuum is an original idea. This idea generalizes fundamental foundations of both classical and relativistic mechanics that study spatiotemporal characteristics of material objects. In our case the scope is generalized to include imaginary conceptual entities. The resulting space-time-constitution framework can be considered as an extension of the traditional Minkowski's space-time framework;
- the vision of defining information as a continuum emerging out from the space-time and the model constitution continuums being considered in the same

context is an original idea that however has an analogy found in Taoist philosophy.

The important result was to demonstrate that in the Triune Continuum Paradigm the conceptualization of the universe of discourse is in agreement with fundamental philosophical and natural science foundations. This demonstration (that can be found in [7]) allows us to rely on the introduced conceptualization and thus to define the set of Tarski's declarative semantics for the basic modeling concepts from the paradigm as not only having the logical consistency in the interpretation, but also being justified as a generalization of scientific experience. And as we explained in the beginning of this section, the definition of this Tarski's declarative semantics set for the limited modeling scope introduced by the conceptualization provided a straightforward logical proof that the resulting limited set of basic modeling concepts is necessary and sufficient for the modeling scope representation.

5.4. Impact of the proposed solutions on the UML terminology

In sections 5.1-5.3 we presented solutions for the three problems of UML metamodel. Thus, unlike the currently existing UML metamodel [9], the alternative metamodel that is shaped by the presented solutions would not cause the problematic effects for the UML terminology (see Section 4.4 of this paper). Moreover, apart from being destitute of the destructive metamodeling features found in the current UML specifications, the alternative metamodel based on the Triune Continuum Paradigm exhibits a concrete constructive potential for realization of different object-oriented terminologies, in particular:

- based on its declarative semantics for the basic modeling concepts it ensures the coherency in interpretations of subjects of modeling interest for any of its conforming terminologies;
- based on the rigorous logical structure of its organization (supported by Russell's theory of types) it ensures:
  - internal consistency of any of its conforming terminologies;
  - unambiguity in application contexts for the concepts of any of its conforming terminologies (e.g. absence of Russell's paradoxes);
- based on its scientific foundations it ensures the theoretical soundness for any of its conforming terminologies.

Thus the proposed metamodel can be useful not only for UML but also for other different modeling frameworks which lack the aforementioned positive features in their current states.

As for UML, the analysis presented in Section 4.4 shows that not only its current terminology does not conform to the metamodel proposed by the Triune Continuum Paradigm, but moreover that the core of the UML terminology in its current state can be considered as undefined. Hence, in any case, UML needs a new terminology (and current efforts of OMG on the definition of the next version of UML confirm this conclusion). Thus if this new terminology will be conformant with the Triune Continuum Paradigm, then UML would benefit from all the aforementioned advantages.

We are aware of multiple propositions (e.g. [1], [2], [13]) defining the new terminology for the next version of UML. In fact any of such propositions can be conformant with the Triune Continuum Paradigm as soon as it obeys the following two requirements (see [7] for the explanations):

- in the definitions of its conceptual framework it should be destitute of self-contradictions;
- it should allow to reserve the set of eight concepts, namely the concepts analogous to: *Object*, *Environment*, *Point in Time*, *Point in Space*, *Time Interval*, *Space Interval*, *Action* and *State*, as corresponding to the paradigm's basic modeling concepts and thus having predefined declarative semantics.

So, because of the intrinsic flexibility of the Triune Continuum Paradigm, any of UML contributors can benefit from the paradigm's metamodel that we presented in this paper. This shows the complementarity of our research results in the relation to the mainstream research on definition of the next version of UML. And because of this complementarity the paradigm-based metamodel is probably the most interesting of our results.

Another result that we realized ([7], [8]) is a concrete object-oriented terminology that conforms to the paradigm and that can serve as an internally consistent solution for the UML terminology. It is the conceptual framework of RM-ODP [3] that was reinforced with the aid of interpretation constraints provided by the Triune Continuum Paradigm. But this is one of many possible solutions and in the general case we don't see any reason to favor it over other terminologies conformant with the Triune Continuum Paradigm. This is why this terminology is not our primary concern in this paper. Though in the particular case of UML there are some additional reasons in favor of our RM-ODP-based terminology, in particular the following facts:

- RM-ODP [3] is an ISO/ITU standard;
- the RM-ODP-based terminology was formalized and presented in a computer-interpretable form ([7], [8]);
- UML specifications mention RM-ODP as a framework that has already influenced UML metamodel architectures ([9] in *Preface: Relationships to Other Models*).

5. Conclusions

In this paper we presented an overview of the Triune Continuum Paradigm [7], reviewing the advantages that the paradigm brings to modelers and its fundamental theoretical foundations. Also we demonstrated the constructive potential that the paradigm may provide for the Unified Modeling Language (UML). In particular, we reviewed three important problems of the UML metamodel. These were the following problems:

- Absence of an explicit structural organization defined for the UML metamodel;
- Absence of formal declarative semantics in the UML metamodel;
- Absence of theoretical justifications for the UML metamodel to represent the modeling scope that is targeted by UML.

As we showed, the Triune Continuum Paradigm proposes concrete solutions for these problems. In particular:

- its proposed metamodel has an internally consistent structure supported by Russell's theory of types [11];
- it defines a kind of Tarski's declarative semantics [12] for the basic modeling concepts, thus it is coherent and unambiguous in the interpretations of subjects of modeling;
- it provides philosophical and natural science foundations to justify that its proposed modeling concepts set is necessary and sufficient to represent its identified modeling scope [7].

So, the paradigm-based solutions define concrete improvements for the current state of UML metamodel, and they can have a constructive influence on the evolution of UML by providing the language designers with the paradigm's logical rigor, its formal presentation and its solid theoretical foundations.

In this paper we also showed that because of the analyzed problems:

- the UML terminology in its current state does not have any explicit reason supporting the terminology introduction;
- the terminology contains self-contradictions and baseless references (references to undefined concepts).

We demonstrated that existing UML terminology doesn't allow for its logically consistent interpretation. The Triune Continuum Paradigm was applied on the RM-ODP conceptual framework realizing a concrete object-oriented terminology that was formalized in a computer-interpretable form [7], [8]. This realization is a possible internally consistent solution for the UML terminology.

The three problems of the UML metamodel and their subsequent problems of the UML terminology clearly demonstrate that at the present time from the theoretical standpoint, UML assimilates itself to the naked Emperor from the famous story: "The Emperor's New Clothes" by Hans Christian Andersen (1805-1875). In the story those who were not able to see the most magnificent cloth of the Emperor were considered as either stupid or incompetent. So everybody, including the Emperor himself, pretended to be able to see the cloth regardless of its nonexistence. This continued until people heard the voice of an innocent child: "But the Emperor has nothing at all on!". Then the people began to whisper to one another what the child had said, till everyone was saying: "But he has nothing on!". The Emperor himself understood that the people were right, but he was not courageous enough to publicly acknowledge the fault. "*So he drew himself up and walked boldly on holding his head higher than before, and the courtiers held on to the train that wasn't there at all.*" – the story ends.

The analogy with UML semantics is straightforward. Our research results show:

- that the UML metamodel does not exhibit some of the practically essential metamodeling features (e.g. an explicit structural organization and formal declarative semantics definition);
- that the UML metamodel does not have any theoretical support to justify its claims for an adequate representation of the identified (by UML) modeling scope;
- that the UML terminology definitions are not even a house of cards that can be ruined by a light wind, but rather the baseless ruins from which it is impossible to reconstruct a single coherent framework.

All these facts are not surprising. UML, being a widely promoted solution that occupied a leading position in modern industrial applications, appeared as an integration of the best software development practices. That is, roughly speaking, it is a result of the vast practical experience: tries, failures and successes that were never theoretically justified. That's why it is somewhat natural that in its current state UML doesn't propose a formal definition of its applications scope, that semantics of UML concepts are contradictory and that the metamodel of UML does not feature a coherency in modeling interpretations. But the fact that it is "somewhat natural", does not mean that it is good. There are many unfortunate practical consequences that are caused by these problems. For example, because of the impossibility to understand the limits of allowed UML applications scope, the adequacy of UML modeling inevitably suffers in those multiple cases when the language is not applicable or when it is applicable with additional constraints.

We acknowledge the comparative practical domination of UML in the current industrial modeling practices. We do not call in question the relatively successful definition of constraints for practical applications of UML

formulated in section 3 ("UML Notation Guide") of UML specifications [9]. But, as we showed, from the theoretical standpoint UML "has nothing at all on!". And obviously, even successfully promoting UML will not eliminate this very important problem. "The Emperor" may keep "walking boldly on holding his head higher than before", pretending that "the most magnificent cloth" is "*very difficult for beginners to understand*" [10] but the theoretical "cloth" for UML will not appear by itself.

The Triune Continuum Paradigm [7], which was presented in Section 2, provides concrete solutions for the problems of UML that were reviewed in this paper. The concreteness of these solutions and the fact that they are formally implemented [7], [8] on the example of RM-ODP (the framework that was mentioned by UML specifications as influential for the UML metamodel architectures, see [9] in *Preface: Relationships to Other Models*) are two strong points that may attract the system modeling community's attention to our research results.

6. References

[1] 3C UML: "*OMG Unified Modeling Language Specification (revised submission)*", Version 2.0.11, Revised Infrastructure Submission, June 2002. http://www.community-ml.org/docs/3C_Infra_Revised.pdf

[2] Clark, T., Evans, A., France, R., Kent, S., Rumpe, B.: "*Response to UML 2.0 Request for Information*", Submitted by the precise UML group, December 1999. http://www.cs.york.ac.uk/puml/papers/RFIResponse.PDF

[3] ISO, ITU.: ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation X.901, X.902, X.903, X.904. "*Open Distributed Processing - Reference Model*". 1995-98.

[4] Einstein, A., Lorentz, H. A., Minkowski, H., Weyl, H.: "*Principle of Relativity*"; a collection of original memoirs on the special and general theory of relativity. Dover Publications, New York, 1952.

[5] Frege, G., 1903, "The Russell Paradox", in Frege, G., *The Basic Laws of Arithmetic,* Berkeley: University of California Press, 1964, pp. 127-143.

[6] Jackson D.: "Alloy: A Lightweight Object Modelling Notation". *ACM Transactions on Software Engineering and Methodology*. Volume 11, Issue 2. April 2002, pp. 256-290.

[7] Naumenko, A.: "*Triune Continuum Paradigm: a paradigm for General System Modeling and its applications for UML and RM-ODP*". Ph.D thesis 2581, Swiss Federal Institute of Technology - Lausanne, June 2002. http://lamswww.epfl.ch/people/naumenko/Naumenko-PhD-Thesis.pdf

[8] Naumenko, A., Wegmann, A., Genilloud, G., Frank, W. F.: "Proposal for a formal foundation of RM-ODP concepts". *Proceedings of ICEIS 2001, WOODPECKER`2001*, J. Cordeiro, H. Kilov (Eds.), Setúbal, Portugal, July 2001, pp. 81-97

[9] OMG: *Unified Modeling Language Specification*. Version 1.4, September 2001.

[10] OMG: *Introduction to OMG's Unified Modeling Language (UML™)*. January 2002, http://www.omg.org/gettingstarted/what_is_uml.htm

[11] Russell, B.: "Mathematical logic as based on the theory of types". *American J. of Mathematics*, 30, 1908, pp. 222-262.

[12] Tarski, A.: "*Logic, Semantics, Meta-mathematics.*" Oxford University Press 1956.

[13] U2 Partners: "*Unified Modeling Language 2.0 Proposal.*" Version 0.671; Revised submission to OMG RFPs ad/00-09-01 and ad/00-09-02, January 2002. http://www.u2-partners.org/artifacts.htm